



국민대학교
소프트웨어학부
C++프로그래밍

결과보고서

프로젝트 명

Snake Game

팀 명

김혜민

Confidential Restricted

Version 1.3

2023-6-18

C++프로그래밍 프로젝트


프로젝트 명	Snake Game
팀 명	김혜민
문서 제목	결과 보고서

Version	1.3
Date	2023-06-18

팀원	김윤희
	신수민
	임혜진

CONFIDENTIALITY/SECURITY WARNING


이 문서에 포함되어 있는 정보는 국민대학교 소프트웨어융합대학 소프트웨어학부 및 소프트웨어학부 개설 교과목 C++프로그래밍 수강 학생 중 프로젝트 “Snake Game”를 수행하는 팀 “김혜민”의 팀원들의 자산입니다. 국민대학교 소프트웨어학부 및 팀 “김혜민”의 팀원들의 서면 허락없이 사용되거나, 재가공 될 수 없습니다.

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	김혜민	
	Confidential Restricted	Version 1.3	2023-6-18

문서 정보 / 수정 내역


Filename	Snake Game
원안작성자	김윤희, 신수민, 임혜진
수정작업자	김윤희, 신수민, 임혜진

수정날짜	대표수정자	Revision	추가/수정 항목	내 용
2023-05-27	신수민	1.0	최초 작성	보고서 틀 및 개요 작성
2023-05-31	김윤희	1.1	내용 수정	추가 및 수정된 개발 내용 추가
2023-06-01	임혜진	1.2	내용 수정	추가 및 수정된 개발 내용 추가
2023-06-19	모두	1.3	마무리	부가요소 작성 및 마무리

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	김혜민	
	Confidential Restricted	Version 1.3	2023-6-18

목 차

1	개요	4
1.1	프로젝트 소개 및 개발 방법	4
1.2	개발 과정	4
1.3	ncurses 라이브러리 소개 및 설치 방법	5
2	개발 내용 및 결과물	7
2.1	목표	7
2.2	개발 내용 및 결과물	11
2.2.1	개발 내용	11
2.2.2	시스템 구조 및 설계도	13
2.2.3	활용/개발된 기술	43
2.2.4	현실적 제한 요소 및 그 해결 방안	44
2.2.5	결과물 목록	45
3	자기평가	46
4	참고 문헌	47
5	부록	48
5.1	사용자 매뉴얼	48
5.2	설치 방법	53

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	김혜민	
	Confidential Restricted	Version 1.3	2023-6-18

1 개요

평가기준 (10점)

프로젝트를 완성하기 위해 사용한 개발 방법을 기술하세요.

또한 사용하고 있는 외부 라이브러리와 해당 라이브러리를 획득/설치하는 방법을 기술하세요.

1.1 프로젝트 소개 및 개발 방법

본 프로젝트는 2023년 1학기 C++ 프로그래밍의 마지막 과제로, C++ 프로그래밍 언어로 ncurses 라이브러리를 사용하여 Snake Game을 구현하는 팀 프로젝트이다. 과제 마일스톤에 따라 1~5단계를 구현했으며, class와 struct를 만들고 여러 기능에 사용되는 멤버 함수들을 정의하였다.

1.2 개발 과정


<역할 분담>

이름	맡은 부분
김윤희	게임 구현, 보고서 작성
신수민	게임 구현, 보고서 작성, 시연영상 제작
임혜진	게임 구현, 보고서 작성

※ 각 메소드별 개발자는 2.2.2에서 기재하였습니다.

<개발 과정>


- 2023. 05. 22.
 - 팀명 정하기
 - 역할 분담하기
- 2023. 05. 24
 - 각 요소들의 색 정하기 (ex: 벽 = 하얀색)
 - 시작 화면에 어떤 메뉴들을 띄울 것인지에 대한 회의 후 구현
 - Ncurses library의 어떤 내장 함수를 사용하여 Map을 구현할 것인지 회의 후 구현
 - 새로 추가할 Item에 대한 아이디어 회의
- 2023. 05. 25.
 - Tick에 대한 어떤 변화를 줄 것인지에 대한 아이디어 회의
 - Map 위에 Snake의 움직임을 어떻게 자연스럽게 줄 것인지에 대한 회의 후 구현

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	김혜민	
	Confidential Restricted	Version 1.3	2023-6-18

- 2023. 05. 27.
 - Wall에 대한 변화는 어떤 방식으로 주는 게 좋을지에 대한 아이디어 회의
 - 시작 화면에서 HELP로 이동했을 때, 자세히 어떤 것에 대한 도움말을 포함할 것인지에 대한 회의 후 구현
 - Item에 Snake가 닿아도 Snake의 길이가 변하지 않아 다같이 구현
 - 보고서 틀 작성
- 2023. 05. 29.
 - Mission Board와 Gate, Magic Gate 통과 카운트 간의 상호작용 구현
 - Score Board와 Item들의 카운트 간의 상호작용 구현
 - Growth Item의 아이콘 변경 제안
- 2023. 05. 31
 - Mission 세부 설정 회의
 - 보고서 시스템 구조 및 설계도 작성
- 2023. 06. 01.
 - Score Board 를 통한 게임 점수 표시 구현
 - Score Board에 새로 추가할 요소 회의 및 구현
 - 모든 Stage를 통과했을 경우 화면 관련 회의
 - 보고서 사용자 메뉴얼 등의 부가요소 작성
- 2023. 06. 18.
 - 전체적인 코드 보완 및 효율적인 알고리즘 모색
 - 각 단계별 영상 촬영 후 업로드
- 2023. 06. 19.
 - github에 업로드 후 협업 설정
 - 보고서 작성 마무리


1.3 ncurses 라이브러리

이 프로젝트에는 ncurses라는 외부 라이브러리가 사용되었다. ncurses는 New Curses의 약자로, curses의 새로운 버전이라는 의미이다. 이 라이브러리는 텍스트 모드에서 Window, Panel, Menu, Mouse, Color 등을 쉽게 사용할 수 있도록 도와 준다. ncurses는 리눅스 라이브러리이며, 최신 버전의 리눅스의 경우 gcc 같은 개발 프로그램을 설치 시 자동으로 설치된다. 운영체제에 따른 설치 방법은

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	김혜민	
	Confidential Restricted	Version 1.3	2023-6-18

다음과 같다.

- Mac
Brew install ncurses
- Ubuntu
sudo apt-get update
sudo apt-get install libncurses5-dev libncursesw5-dev

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	김혜민	
	Confidential Restricted	Version 1.3	2023-6-18

2 개발 내용 및 결과물

2.1 목표

작성요령 (10점)

프로젝트의 목표를 기술하세요. 각 단계별 목표를 구체적으로 쓰세요.

적용단계	내용	적용 여부
1단계	Map의 구현	적용
2단계	Snake 표현 및 조작	적용
3단계	Item 요소의 구현	적용
4단계	Gate 요소의 구현	적용
5단계	점수 요소의 구현	적용
6단계	게임 PLAY, HELP, EXIT 구현	추가적용
비고	Item, Gate, Score Board에 대한 추가 구현	추가적용

1단계 : Map의 구현


1단계의 주 목표는 Ncurses Library 함수들을 사용하여 2차원 배열로 된 Map을 Game 화면으로 표시하는 프로그램을 완성하는 것입니다.

- stage 를 4단계로 구분하고, stage가 증가할수록 난이도도 증가하는 방향으로 구현한다.
- Map의 세로길이를 가로길이를 설정한다.
- wall과 Immune wall 을 잘 구분해야 한다.

2단계 : Snake 표현 및 조작

2단계의 주 목표는 1단계에서 구현한 맵 위에 Snake를 표시하고, 화살표를 입력받아 방향에 맞게 Snake가 움직이도록 프로그램을 완성하는 것입니다.

- Snake는 규칙 #1을 준수해야 한다.
 - 방향키는 사용자가 직접 정한다.
 - Snake 는 진행 방향의 반대 방향으로 이동할 수 없다. 반대 방향 키를 입력할 경우 게임이 종료된다. (head 가 진행 방향이며, tail 방향으로 이동할 수 없다)

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	김혜민	
	Confidential Restricted	Version 1.3	2023-6-18

- 진행 방향과 같은 방향키 입력은 무시한다. (head 방향 이동은 무시한다)

- (추가사항) Tick에 대한 변화 주기

Snake 는 일정 시간(틱)에 의해 이동한다.

스테이지가 올라갈 때마다 틱이 빨라지는 방식으로 변화를 주어 난이도가 올라갈 수 있게 하였다.

• 이외의 실패 조건

- Snake는 자신의 body를 통과할 수 없다.

- Snake는 벽(wall)을 통과할 수 없다.

3단계 : Item 요소의 구현

3단계의 주 목표는 Map 위에 Growth Item과 Poison Item을 출현하도록 프로그램을 완성하는 것 입니다.

• 게임 규칙 #2을 준수해야 한다.

- Snake 의 이동 방향에 item 이 놓여 있는 경우 snake 가 item 을 획득한다.

- Growth item 의 경우 몸의 길이가 진행 방향으로 1 증가한다.

- Poison item 의 경우 몸의 길이가 1 감소한다.

- (추가사항) Item 종류에 대한 추가 (Speed Slow Item)

Speed Slow Item 의 경우 스피드가 현재 속도에서 감소된다.

- 몸의 길이가 3 보다 작아지면 게임이 종료된다.

• Growth item 과 Poison item 의 출현

- Item 은 snake body 가 있지 않은 임의의 위치에 출현해야 한다.


- 출현 후 일정 시간(5 초)가 지나면 사라지고, 랜덤한 다른 위치에 나타나야한다.

- 동시에 출현할 수 있는 item 의 개수는 3 개로 제한한다.

• Growth Item 과 Poison Item 을 Map 배열에 표현할 때 값을 정한다.


• 화면 상에 표현 시, 색이나 기호를 달리하여 구분할 수 있도록 한다.

4단계 : Gate 요소의 구현

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	김혜민	
	Confidential Restricted	Version 1.3	2023-6-18

4단계의 주 목표는 Map의 Wall 위의 임의의 위치에 한 쌍의 Gate가 출현할 수 있도록 변경하고, 각 Gate를 Snake가 통과할 수 있도록 프로그램을 완성하는 것입니다.

- 게임 규칙 #3, #4, #5 를 준수해야 한다.
 - Gate 는 두 개가 한 쌍이다.
 - Gate 는 겹치지 않는다.
 - Gate 는 벽(wall)위의 임의의 위치에서 나타난다.
 - Gate 에 snake 가 진입하면, 다른 Gate 로 진출한다.
 - Gate 는 한번에 한 쌍만 나타난다.
 - Gate 에 snake 가 진입 중인 경우, Gate 는 사라지지 않아야 하며, 동시에 다른 위치에서 Gate 가 나타나면 안 된다.
- Gate 가 나타나는 벽이 가장자리에 있을 때
 - 항상 Map 의 안쪽 방향으로 진출한다.(고정)
 - 상단 벽 => 아래 방향
 - 하단 벽 => 위 방향
 - 좌측 벽 => 오른쪽 방향
 - 우측 벽 => 왼쪽 방향
- Gate 가 나타나는 벽이 Map 의 가운데 있을 때 다음의 순서로 진출
 - 진입 방향과 일치하는 방향이 우선
 - 진입 방향의 시계 방향으로 회전하는 방향
 - 진입 방향의 역시계 방향으로 회전하는 방향
 - 진입 방향과 반대 방향
- 화면 상에 표현 시, Gate 는 Wall 과 구분될 수 있도록 한다.


 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	김혜민	
	Confidential Restricted	Version 1.3	2023-6-18

- Wall(Immune Wall 포함)과 Gate 를 Map 배열에 표현할 때 값을 결정한다.
- (추가 사항) Wall에 대한 변화 추가 사항
 - Wall의 임의의 위치에 Magic Wall이 출현하는 기능을 추가했다. Magic Gate를 통과하면 Magic Exit으로 나오게 되는데 Magic Exit은 화면에 표시되지 않는다. Magic Gate 통과 후 진출하는 순서 등 다른 규칙은 Gate와 동일하다.

5단계 : 점수 요소의 구현

5단계의 주 목표는 우측에 게임 점수를 표시하는 화면을 구성하는 것입니다.

- score board 와 mission board 를 각자 다른 윈도우로 구현하여 우측에 띄운다.
- 게임 점수는 게임 규칙 #6 을 준수한다.
 - 게임 중 몸의 최대 길이 계산 (B : 현재 길이/최대 길이)
 - 게임 중 획득한 Growth Item 의 수 (+)
 - 게임 중 획득한 Poison Item 의 수 (-)
 - 게임 중 Gate 사용 횟수 (G)
- mission 은 구성된 Map 의 정의에 고정 값을 주는 방식으로 수행한다.
- stage 마다 Mission 을 달성하면 다음 Map 으로 진행하도록 프로그램을 완성한다.
- Stage 는 최소 4 개로 구성하고, 각 Stage 의 Map 은 서로 달라야 한다.
- (추가 사항) Score Board 에 대한 변경
 - 3단계 추가 사항으로 구현한 Speed Slow도 Score Board에 표시되도록 요소를 추가했다. Speed Slow Item을 먹은 개수를 Score Board에서 확인할 수 있다.

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	김혜민	
	Confidential Restricted	Version 1.3	2023-6-18

2.2 개발 내용 및 결과물

2.2.1 개발 내용

작성요령 (10점)

프로젝트의 수행의 내용을 구체적으로 기술한다. 세부 목표별로 어떤 결과를 어떤 방법으로 달성하였는지를 자세히 기술한다.

1. Map의 구현


1. 큰 틀에서 3차원 stage 배열을 생성한다.
2. stage에 따라 항목별 인덱스를 달리 저장하고 stage 시작 시마다 2차원 map 배열에 값을 복사하여 동작에 따라 출력한다.
3. map은 0.1초 간격으로 자동 refresh되며, 조건에 따라 실시간으로 값이 변경되어 출력되도록 하였다.
4. 출력 시에 각 항목별 시각적 구분을 명확히 하기 위해, 색상을 달리 지정하였다.

2. snake 표현 및 조작

1. struct를 이용하여 뱀의 머리 부분과 몸 부분의 객체를 생성하여 꼬리 부분부터 이어주었다. (생성)
2. 꼬리 부분부터 이어진 구조체의 x,y좌표를 자신의 x,y로 바꾸어 출력하면서 이동하였다.
3. 마지막 머리부분은 방향을 알려주는 변수인 dir을 이용하여 다음 좌표를 정해 주었다.

3. Item 요소의 구현

1. map 배열에서 벽이 아닌 좌표 값과 아이템의 번호를 랜덤으로 입력받아 struct 객체를 생성하여 배열에 출력하였다.
2. Growth 아이템은 객체를 하나 더 생성하여 이어붙였다.
3. Poison 아이템은 꼬리를 가리키는 포인터를 한 칸 이동시킨 후, 마지막 객체를 삭제하였다.
4. SpeedSlow 아이템을 생성하여 해당 아이템을 먹을 시 현재 속도가 감소하도록 하였다.


 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	김혜민	
	Confidential Restricted	Version 1.3	2023-6-18

4. Gate 요소의 구현

1. random 함수를 이용하여 어떤 벽에 생성할 것인지 입력받은 후
2. 무작위 숫자를 결정한 벽에 대입하여 일반 게이트와 매직게이트를 생성하였다.
(게이트 변수에 x,y값을 저장하였다.)
3. 머리가 게이트를 만났을 때 다른 게이트의 좌표값+-1으로 이동시켰다.

5. 점수/미션 요소의 구현

1. Snake Length(현재 뱀의 길이), Growth item(증가 아이템 획득 횟수), Poison item(감소 아이템 획득 횟수), Slow Item(속도 감소 아이템 획득), Gate(일반 혹은 Magic 게이트 진입 횟수)의 각 현재 점수를 stat 배열로써 다루고 출력하며, 마찬가지로 미션 점수도 statMission 배열로써 다루고 난수를 발생하게 하여 각각 초기화 후에 출력한다.
2. setMission(미션지정) 함수가 동작할 때, memset 함수를 활용하여 stat과 statMission, chkMission 배열을 각각 크기만큼 모든 인덱스의 값을 0으로 초기화한다.
3. 스코어 보드에서 해당 스탯이 증가 또는 감소할 때마다 그것에 해당하는 stat의 값을 증가, 감소시킨다.
4. 미션에 해당하는 숫자를 달성할 경우, chkMission 배열의 해당 인덱스에 'v'를 저장하고 출력한다.
5. 각각의 값들은 스코어/미션 보드의 양식에 맞게 정확히 출력한다.
6. 미션의 달성/미달성의 경우, 미션 보드의 각 항목의 우측 괄호안에 'v'표시로 체크/해제 한다.

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	김혜민	
	Confidential Restricted	Version 1.3	2023-6-18

2.2.2 시스템 구조 및 설계도

작성요령 (30점)

프로젝트의 각 세부 목표의 주요 기능(알고리즘 등)에 대해서 기술한다. 세부 목표별로 수정한 프로그램 소스 파일을 나열하고, 해당 파일에서 세부 목표를 달성하기 위해 작성한 클래스/함수에 대해 나열하고, 각 요소에 대해 간략한 설명을 작성한다. 또한 각 요소의 개발자를 명시한다.

[Stage.h, Stage.cpp]

- void screenLock(); 개발자 : 신수민

```
void Stage::screenLock()
{
    cout << "\e[3;240;120t";
    cout << "\e[8;40;120t";
    system("resize -s 40 120");
    y = 40, x = 120;
    mvprintw(y - 1, 0, "SnakeGame ver. 1.0");
    sizeY = y / 1.5,
    sizeX = x / 1.5,
    startY = y / 2 - sizeY / 2,
    startX = x / 2 - sizeX / 2,
    desSizeY = sizeY - 6,
    desSizeX = sizeX - 6,
    desStartY = startY + 3,
    desStartX = startX + 3,
    txtLines = 26,

    hidTxtLen = txtLines - desSizeY > 0 ? txtLines - desSizeY : 0,
    scrollbarLen = desSizeY - hidTxtLen;
}
```

Snake 게임의 화면을 잠금(lock) 상태로 설정하고, 터미널 창의 크기를 조절하며, 텍스트 영역의 크기와 위치를 계산해주는 함수이다. 이를 통해 게임 화면이 적절한 크기로 표시되고, 텍스트 영역이 올바른 위치에 표시된다.


- string menu(); 개발자 : 신수민

```
string Stage::menu()
{
```



```
clear();

screenLock();
curs_set(0);
string txt[4];
txt[0] = "[ SNAKE GAME ]";
int focus = menuLastFocus;
level = 0;
while (1)
{
    if (!focus)
        focus = 300;
    txt[1] = "PLAY";
    txt[2] = "HELP";
    txt[3] = "EXIT";
    attron(COLOR_PAIR(10));
    mvprintw(y / 2 - 2, x / 2 - txt[0].length() / 2, txt[0].c_str());
    attroff(COLOR_PAIR(10));
    for (int i = 1; i < sizeof(txt) / sizeof(txt[0]); i++)
    {
        if (i == abs(focus % 4 + 1))
        {
            attron(COLOR_PAIR(11));
            mvprintw(y / 2 + i, x / 2 - (txt[i].length() / 2), txt[i].c_str());
            attroff(COLOR_PAIR(11));
        }
        else
            mvprintw(y / 2 + i, x / 2 - (txt[i].length() / 2), txt[i].c_str());
    }
    switch (getch())
    {
        case UP:
            focus--;
            break;
        case DOWN:
            focus++;
            break;
        case ENTER:
            menuLastFocus = focus;
            return txt[abs(focus % 4 + 1)];
    }
}
```

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	김혜민	
	Confidential Restricted	Version 1.3	2023-6-18

```

return NULL;
}

```

이 함수는 게임의 퀄리티를 위해 추가적으로 텍스트 기반의 사용자 인터페이스를 구현한 것이며, 메뉴 내에서 사용자의 키보드 입력에 따라 화면을 업데이트 한다. 화면 초기화, 스크린 잠금 활성화, 커서 숨김, 메뉴의 제목 색상 등이 정의되고 실행된다.

- void play(); 개발자 : 신수민, 김윤희, 임혜진

```


void Stage::play()
{
    screenLock();
    setMap();
    int n;
    for (int i = 0; i < STAGE_NUM; i++)
    {
        timeoutMs = speedMs[speed - 1];
        msTime = n = 0;
        dir = LEFT;
        copyMap(i);
        setMission();
        makeSnake();
        appearGate();
        appearMagicGate();
        drawMap();
        while (1)
        {
            switch (getch())
            {
                case LEFT:
                    dir = LEFT;
                    break;
                case UP:
                    dir = UP;
                    break;
                case RIGHT:
                    dir = RIGHT;
                    break;
                case DOWN:
                    dir = DOWN;

```



```
        break;

    case PAUSE:
        alert(y / 2 - 4, x / 2 - 34, "Press 'r' to play!", TRUE);
        while (1)
        {
            if (getch() == RESUME)
                break;
        }
        break;
    case ESC:
        endwin();
        return;
    }
    moveSnake();
    if (chkEnter)
    {
        if (++n >= stat[0])
        {
            disappearGate();
            disappearMagicGate();
            appearGate();
            appearMagicGate();
            n = 0;
            chkEnter = FALSE;
        }
    }
    if (++msTime % (msDiv[speed - 1] * 5) == 0)
    {
        disappearItem();
        appearItem();
    }
    if (stat[0] < 3)
        gameOver();
    if (isMissionClear())
    {
        alert(y / 2 - 4, x / 2 - 27, "Stage Clear!", FALSE);
        speed++;
        break;
    }
    if (checkGameOver())
    {
        alert(y / 2 - 4, x / 2 - 25, "Game Over!", FALSE);
        return;
    }
}
```


 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	김혜민	
	Confidential Restricted	Version 1.3	2023-6-18

```

    }

    drawMap();
    timeout(timeoutMs);
  }
  level++;
}
endwin();
}

```

이 함수는 SnakeGame을 실제 플레이하는 로직이다. 함수의 주요 동작으로는

1. 화면을 잠그고 게임 맵을 설정한다.
2. 게임 플레이의 스피드를 설정한다.
3. 각각의 stage에 대한 다음의 동작들을 수행한다
 - 맵 복사, 스피드 설정, 뱀 생성, 게이트(일반, Magic) 표시
 - 사용자의 입력을 체크하여 각 방향 키에 맞는 뱀의 이동
 - 일시중지 버튼을 통해 중지, 'Y'키를 이용한 재시작
 - ESC 키를 누르면 게임 종료
 - 뱀이 게이트에 통과 횟수 기억
 - 정해진 간격으로 아이템 생성 및 소멸
 - 뱀의 길이가 3보다 작을 시 종료
 - 게임 종료 조건을 체크하여 함수 종료
4. 모든 stage를 완료하면 endwin() 함수를 호출하여 게임을 종료한다.

즉, 이 메서드는 게임의 기본적인 흐름과 상호작용을 관리하며 사용자 입력에 따른 뱀의 움직임, 아이템 생성과 소멸, 게임 오버 조건 등을 처리한다.

- void help(); 개발자 : 신수민

```

void Stage::help()
{
    screenLock();
    int ySize = 0, yScroll = 0;
    while (1)
    {
        manual = newwin(sizeY, sizeX, startY, startX);
        description = newwin(desSizeY, desSizeX, desStartY, desStartX);
        scrollBar = newwin(scrollBarLen, 2, desStartY + yScroll, startX + sizeX - 6);
        wattron(manual, COLOR_PAIR(10));
    }
}

```



```
box(manual, 0, 0);

mvwprintw(manual, 0, sizeX / 2 - manualTitle.length() / 2, "%s",
manualTitle.c_str());

wattroff(manual, COLOR_PAIR(10));

mvwprintw(description, 0 + ySize,
sizeX / 2 - menuTitle.length() / 2 - 3, "%s", menuTitle.c_str());

for (int i = 0; i < sizeof(menuTxt) / sizeof(menuTxt[0]); i++)
    mvwprintw(description, 2 + (i * 2) + ySize, sizeX / 2 -
menuTxt[2].length() / 2 - 3, "%s", menuTxt[i].c_str());

mvwprintw(description, 11 + ySize,
sizeX / 2 - shorTitle.length() / 2 - 3, "%s", shorTitle.c_str());

for (int i = 0; i < sizeof(shorTxt) / sizeof(shorTxt[0]); i++)
    mvwprintw(description, 13 + (i * 2) + ySize, sizeX / 2 -
shorTxt[6].length() / 2 - 3, "%s", shorTxt[i].c_str());

if (txtLines >= desSizeY)
{
    wattron(scrollBar, COLOR_PAIR(10));
    box(scrollBar, 0, 0);
    wattroff(scrollBar, COLOR_PAIR(10));
}
refresh();
wrefresh(manual);
wrefresh(description);
wrefresh(scrollBar);

RE:
switch (getch())
{
case UP:
    if (yScroll)
        yScroll--;
    else
        goto RE;
    if (ySize)
        ySize++;
}
```



```
        break;
    case DOWN:
        if (yScroll < desSizeY - scrollBarLen)
            yScroll++;
        else
            goto RE;
        if (ySize > desSizeY - txtLines && txtLines > desSizeY)
            ySize--;
        break;
    case ESC:
        return;
    }
}
```

이 함수는 SnakeGame의 도움말 화면을 출력한다. 함수의 주요 동작으로는

1. 화면 잠금을 설정한다.
2. 사용자가 스크롤 할 수 있는 도움말 텍스트를 출력하기 위한 무한 루프가 시작한다. 루프 내에서는 다음과 같은 작업이 수행된다.
 - 도움말 화면을 위한 새 윈도우 생성, 스크롤 바를 위한 별도의 윈도우 생성.
 - 도움말 화면에 제목과 설명(메뉴 설명 및 단축키 설명)을 출력.
3. 사용자의 입력을 받아 처리한다.
 - 사용자가 위쪽 키를 누르면 위로 스크롤
 - 사용자가 아래쪽 키를 누르면 아래로 스크롤
 - 사용자가 ESC를 누르면 도움말 화면이 종료되고 이전 화면으로 되돌아감.

즉, 이 함수는 사용자가 게임의 조작법이나 메뉴에 대한 정보를 쉽게 이해할 수 있도록 도움말 화면을 제공한다.

- void setMap(); 개발자 : 김윤희

```
void Stage::setMap()
{
    int i, j, k;
    stage = new int **[STAGE_NUM];
    for (i = 0; i < STAGE_NUM; i++)
    {
        stage[i] = new int *[MAP_ROW];
    }
}
```



```
for (j = 0; j < MAP_ROW; j++)

{
    stage[i][j] = new int[MAP_COL];
}

}

for (i = 0; i < STAGE_NUM; i++)
{
    for (j = 0; j < MAP_ROW; j++)
    {
        for (k = 0; k < MAP_COL; k++)
        {
            if (!j || !k || j == ROW_END || k == COL_END)
                stage[i][j][k] = WALL;
            else
                stage[i][j][k] = EMPTY;
        }
    }
    stage[i][0][0] = IMMUNE_WALL;
    stage[i][0][COL_END] = IMMUNE_WALL;
    stage[i][ROW_END][0] = IMMUNE_WALL;
    stage[i][ROW_END][COL_END] = IMMUNE_WALL;
    if (i == 1)
    {
        for (int z = 10; z < 40; z++)
            stage[i][7][z] = WALL;
        for (int z = 10; z < 40; z++)
            stage[i][MAP_ROW - 7][z] = WALL;
    }
    if (i == 2)
    {
        for (int z = 5; z < 20; z++)
            stage[i][z][MAP_COL - 15] = WALL;
        for (int z = 5; z < 20; z++)
```



```
        stage[i][z][15] = WALL;

    }

    if (i == 3)
    {
        for (int z = 10; z < 40; z++)
        {

            if (z > 22 && z < 27)
                continue;

            stage[i][7][z] = WALL;

        }
        for (int z = 10; z < 40; z++)
        {
            if (z > 22 && z < 27)
                continue;

            stage[i][MAP_ROW - 7][z] = WALL;


        }
        for (int z = 5; z < 20; z++)
        {
            if (z > 10 && z < 14)
                continue;

            if (stage[i][z][MAP_COL - 15] == WALL)
                stage[i][z][MAP_COL - 15] = IMMUNE_WALL;
            else
                stage[i][z][MAP_COL - 15] = WALL;

        }
        for (int z = 5; z < 20; z++)
        {
            if (z > 10 && z < 14)
                continue;

            if (stage[i][z][15] == WALL)
                stage[i][z][15] = IMMUNE_WALL;
            else
                stage[i][z][15] = WALL;

        }
    }
}
```

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	김혜민	
	Confidential Restricted	Version 1.3	2023-6-18



이 함수는 SnakeGame의 stage에 대한 맵을 설정한다. 함수의 주요 동작으로는

1. STAGE_NUM 수 만큼의 맵을 생성한다. 각 맵은 행과 열로 구성된 2차원 배열이다.
2. 각 stage의 맵에 대해 외곽을 벽('wall')로 설정하고, 내부를 빈 공간으로 설정한다. 또한 맵의 네 모서리는 immune_wall로 설정하여 뱀이 이 벽을 통과하지 못하도록 한다.
3. 2번 stage의 경우, 맵 중간에 2개의 수평 벽을 추가로 생성한다.
4. 3번 stage의 경우, 2번과 3번 stage에서 생성한 벽들을 조합하여 맵을 추가한다. 일부 벽을 삭제하여 통로를 만들고 immune_wall을 추가로 설정한다.


즉, 이 함수는 게임의 각 stage가 다른 장애물 레이아웃을 가지도록 하여, 게임 난이도를 조절하는데 사용된다.

- void copyMap(int nStage) 개발자 : 김윤희

```
void Stage::copyMap(int nStage)
{
    map = new int *[MAP_ROW];
    for (int i = 0; i < MAP_COL; i++)
        map[i] = new int[MAP_COL];
    for (int i = 0; i < MAP_ROW; i++)
    {
        for (int j = 0; j < MAP_COL; j++)
            map[i][j] = stage[nStage][i][j];
    }
}
```

이 함수는 지정된 stage의 맵을 현재 맵으로 복사하는 역할을 한다. 주요 동작으로는

1. MAP_ROW행을 가진 2차원 맵을 새로 생성하고 이는 현재 stage맵을 저장할 용도로 사용된다.
2. 지정된 stage[nStage]를 순회하면서, 각 위치의 값을 map의 동일한 위치에 복사한다.

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	김혜민	
	Confidential Restricted	Version 1.3	2023-6-18

즉, 이 함수를 호출하면 맵이 복사되며 이후의 게임 진행은 이 맵에서 이루어지게 된다. 선택된 stage의 원본 맵을 보존하면서, 게임 진행에 필요한 변경 사항을 맵에 반영하기도 한다.

- void drawMap(); 개발자 : 김윤희

```
void Stage::drawMap()
{
    game = newwin(MAP_ROW, MAP_COL,
                  y / 2 - MAP_ROW / 2, x / 2 - (MAP_COL / 2 + 16));
    for (int i = 0; i < MAP_ROW; i++)
    {
        for (int j = 0; j < MAP_COL; j++)
        {
            int index = map[i][j];
            wattron(game, COLOR_PAIR(index));
            mvwaddch(game, i, j, itemIndex[index]);
            wattroff(game, COLOR_PAIR(index));
        }
        printw("\n");
    }

    score = newwin(19, 30, y / 2 - (MAP_ROW / 2 + 4), x / 2 + MAP_COL / 2 - 7.4);
    wattron(score, COLOR_PAIR(10));
    box(score, 0, 0);
    mvwprintw(score, 0, 10, "[ SCORE ]");
    wattroff(score, COLOR_PAIR(10));
    mvwprintw(score, 3, 5, "Snake Length: %d / %d", stat[0], SNAKE_MAX_LENGTH);
    mvwprintw(score, 6, 5, "Growth Items: %d", stat[1]);
    mvwprintw(score, 9, 5, "Poison Items: %d", stat[2]);
    mvwprintw(score, 12, 5, "Slow Items: %d", stat[4]);
    mvwprintw(score, 15, 5, "Gate: %d", stat[3]);

    mission = newwin(16, 30, y / 2 - (MAP_ROW / 2 + 4) + 19, x / 2 + MAP_COL / 2 - 7.4);
    wattron(mission, COLOR_PAIR(10));
    box(mission, 0, 0);
    mvwprintw(mission, 0, 9, "[ MISSION ]");
    wattroff(mission, COLOR_PAIR(10));

    mvwprintw(mission, 3, 5, "Snake Length: %d ( %c )", statMission[0], chkMission[0]);
}
```



```
mvwprintw(mission, 6, 5, "Growth Items: %d ( %c )", statMission[1],
chkMission[1]);

mvwprintw(mission, 9, 5, "Poison Items: %d ( %c )", statMission[2],
chkMission[2]);

mvwprintw(mission, 12, 5, "Gate: %d ( %c )", statMission[3], chkMission[3]);

info = newwin(4, 15, y / 2 - (MAP_ROW / 2 + 4), x / 2 + MAP_COL / 2 - 47.4);
mvwprintw(info, 0, 1, "[ STAGE %d/%d ]", level + 1, STAGE_NUM);

mvwprintw(info, 2, 3, "< %02d:%02d >", msTime / (msDiv[speed - 1] * 60), (msTime /
msDiv[speed - 1]) % 60);

refresh();
wrefresh(info);
wrefresh(game);
wrefresh(score);
wrefresh(mission);
}
```

이 함수는 게임 창을 그리고 현재 게임 상태를 표시하는 역할을 한다. 주요 동작으로는

1. 새로운 게임 창 game을 만든다. 이 창은 맵을 그릴 위치를 정의한다.
2. score 창을 만들고, 현재 점수를 그린다. 이 창은 게임 점수를 표시하는 데 사용되며 B, +, -, G에 대한 점수가 표시된다.
3. mission창을 만들고 현재 미션 상태를 나타낸다. 미션 창에는 B, +, -, G에 대한 미션의 목표 수치와 현재 상태가 표시된다.
4. info창을 만들고, 현재 게임 정보를 그린다. 이 창에는 현재 stage와 전체 stage 수, 그리고 경과시간이 표시된다.
5. refresh()와 wrefresh() 함수를 사용하여 모든 변경 사항을 화면에 적용한다.

즉, 함수를 호출하면 게임 창, 점수 창, 미션 창, 정보 창이 갱신되며, 사용자는 최신 게임 상태를 확인할 수 있다.

- void appearItem(); 개발자 : 김윤희

```
void Stage::appearItem()
{
    int appearNum = rand() % 3 + 1;
    for (int i = 0; i < appearNum; i++)
    {
        int itemType = rand() % NUMBER_OF_ITEMS + 5; // ITEM types should be
consecutive integers.
    }
}
```




```
while (1)
{
    int y = rand() % (MAP_ROW - 2) + 1;
    int x = rand() % (MAP_COL - 2) + 1;
    if (map[y][x] == EMPTY &&
        map[y][x - 1] != GATE && map[y][x + 1] != GATE &&
        map[y + 1][x] != GATE && map[y - 1][x] != GATE &&
        map[y][x - 1] != MAGIC_GATE && map[y][x + 1] != MAGIC_GATE &&
        map[y + 1][x] != MAGIC_GATE && map[y - 1][x] != MAGIC_GATE)
    {
        map[y][x] = itemType;
        itemPos.push_back(make_pair(y, x));
        break;
    }
}
```

이 함수는 게임 스테이지에서 아이템을 등장시키는 역할을 한다. 아이템의 등장 횟수와 타입, 생성 위치를 받을 때는 난수를 생성하는데, 이때 rand() 함수를 이용한다. 이 함수는 <cstdlib>에 정의되어 있다.

1. 아이템의 등장 횟수

appearNum이라는 int형 변수는 1부터 3까지 랜덤한 숫자를 가진다. 이 변수만큼 반복해서 아이템이 등장한다.

2. 아이템의 타입

itemType이라는 int형 변수를 통해 item의 타입(GROWTH_ITEM or POISON_ITEM or SPEED_SLOW) 이 정해진다.

3. 아이템의 생성 위치

랜덤한 숫자로 좌표값을 받는다. 좌표값이 나타내는 자리가 비었고, 주변에 GATE가 없을 경우, 그 위치는 아이템을 놓을 수 있는 유효한 위치이므로 해당 위치에 아이템을 생성한다. 유효한 위치를 찾지 못했다면, 다시 좌표값을 받는다. 아이템을 생성한 후에는 나중에 disappearItem() 함수에서 아이템을 삭제하기 위해, 생성 위치를 벡터 배열에 저장해둔다.

- void appearGate(); 개발자 : 임혜진

```
void Stage::appearGate()
{
    int n, y, x;
    for (int i = 0; i < 2; i++)
    {
        while (1)
        {
            n = rand() % (!level ? 4 : 5);
            y = rand() % (MAP_ROW - (i?3:2)) + (i?2:1);
            x = rand() % (MAP_COL - (i?3:2)) + (i?2:1);
            switch (n)
```




```
{
    case 0:
        y = 0;
        break;
    case 1:
        x = 0;
        break;
    case 2:
        x = COL_END;

        break;
    case 3:
        y = ROW_END;
        break;
    case 4:
        while (1)
        {
            x = rand() % 30 + 10;
            y = rand() % 15 + 5;
            if (map[y][x] == WALL)
                break;
        }
    }
    if (map[y][x] == WALL)
    {
        map[y][x] = GATE;
        gatePos.push_back(make_pair(y, x));
        break;
    }
}
if (i == 0)
    gate1 = new Something(y, x, GATE);
if (i == 1)
    gate2 = new Something(y, x, GATE);
}
}
```

이 함수는 게임 스테이지에 게이트를 등장시키는 역할을 한다. 2개의 게이트를 생성하기 위해 게이트의 종류와 위치를 정하는 아래의 과정은 두 번 반복된다.

1. 게이트의 종류

rand() 함수를 이용하여 int형 변수 n에 난수를 받는다. 이에 의해 게이트의 종류가 정해진다.

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	김혜민	
	Confidential Restricted	Version 1.3	2023-6-18

2. 게이트의 위치

rand()함수를 이용하여 int형 변수 y와 x에 난수를 받는다. n에 의해 y와 x가 달라질 수 있다. 이에 의해 게이트가 출현할 위치가 정해진다. WALL인 위치만 게이트를 놓을 수 있는 유효한 위치이다. 위치가 유효하다면 해당 위치에 게이트를 출현시킨다. 출현 시킨 게이트는 gate1 혹은 gate2에 저장된다.

- void appearMagicGate(); 개발자 : 임혜진

```
void Stage::appearMagicGate()
{
    int n, y, x;

    for (int i = 0; i < 2; i++)
    {
        while (1)
        {
            n = rand() % (!level ? 4 : 5);
            y = rand() % (MAP_ROW - (i?3:2)) + (i?2:1);
            x = rand() % (MAP_COL - (i?3:2)) + (i?2:1);
            switch (n)
            {
                case 0:
                    y = 0;
                    break;
                case 1:
                    x = 0;
                    break;
                case 2:
                    x = COL_END;
                    break;
                case 3:
                    y = ROW_END;
                    break;
                case 4:
                    while (1)
                    {
                        x = rand() % 30 + 10;
                        y = rand() % 15 + 5;
                        if (map[y][x] == WALL && map[y][x] != GATE)
                            break;
                    }
            }
        }
        if(i==0){
            if (map[y][x] == WALL && map[y][x] != GATE )
```



```
{
    map[y][x] = MAGIC_GATE;
    magicGatePos.push_back(make_pair(y, x));
    break;
}

}

if(i==1){
    if (map[y][x] == WALL && map[y][x] != GATE && map[y][x] != MAGIC_GATE)
    {
        map[y][x] = MAGIC_EXIT;

        magicGatePos.push_back(make_pair(y, x));
        break;
    }
}


if (i == 0)
    magicGate1 = new Something(y, x, MAGIC_GATE);
if (i == 1)
    magicGate2 = new Something(y, x, MAGIC_EXIT);
}
}
```

Magic Gate 역시 appearGate 메소드와 거의 유사하게 동작한다. Magic Gate와 Magic Exit는 벽 중에서도 게이트가 아닌 위치에서만 출현할 수 있다. Magic Gate는 노란색으로, Magic Exit는 벽 색깔과 똑같이 화면에 표시되어 어디로 나올지 알 수 없게 구현하였다.

- void disappearItem(); 개발자 : 김윤희

```
void Stage::disappearItem()
{
    for (auto item : itemPos)
    {
        if (map[item.first][item.second] == GROWTH_ITEM ||
            map[item.first][item.second] == POISON_ITEM || map[item.first][item.second] ==
            SPEED_SLOW)
            map[item.first][item.second] = EMPTY;
    }
    itemPos.clear();
}
```

이 함수는 게임 스테이지에서 아이템을 제거하는 역할을 한다.

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	김혜민	
	Confidential Restricted	Version 1.3	2023-6-18

‘itemPos’ 벡터에 생성할 때 저장해두었던 아이템의 위치를 순회하면서 해당 위치의 ‘map’ 배열 요소가 ‘GROWTH_ITEM’, ‘POISON_ITEM’, ‘SPEED_SLOW’인 경우에만 해당 위치를 ‘EMPTY’로 변경하여 아이템을 제거한다. 마지막으로 ‘itemPos’ 벡터를 비워서 모든 아이템의 위치 정보를 제거한다.

- void disappearGate(); 개발자 : 임혜진

```
void Stage::disappearGate()
{
    for (auto gate : gatePos)
    {
        if (map[gate.first][gate.second] == GATE)
            map[gate.first][gate.second] = WALL;
    }
    gatePos.clear();
}
```

이 함수는 게임 스테이지에서 게이트를 제거하는 역할을 한다. ‘gatePos’ 벡터에 생성할 때 저장해두었던 게이트의 위치를 순회하면서 해당 위치의 ‘map’ 배열 요소가 ‘GATE’인 경우에만 해당 위치를 다시 WALL로 변경한다. 마지막으로 ‘gatePos’ 벡터를 비워서 모든 게이트 위치 정보를 제거한다.

- void disappearMagicGate(); 개발자 : 임혜진

```
void Stage::disappearMagicGate()
{
    for (auto gate : magicGatePos)
    {
        if (map[gate.first][gate.second] == MAGIC_GATE)
            map[gate.first][gate.second] = WALL;
        if (map[gate.first][gate.second] == MAGIC_EXIT)
            map[gate.first][gate.second] = WALL;
    }
    magicGatePos.clear();
}
```

이 함수는 게임 스테이지에서 Magic Gate를 제거하는 역할이다. disappearMagicGate와 거의 비슷하게 동작한다.

- void makeSnake(); 개발자 : 김윤희



```
void Stage::makeSnake()
{
    stat[0] = 3;
    int row = 13;
    int col = 26;

    Bam = new Something(row, col--, SNAKE_BODY);
    Something *p = new Something(row, col--, SNAKE_BODY);
    Bam->link = p;
    p = new Something(row, col--, SNAKE_HEAD);
    Bam->link->link = p;
    map[Bam->y][Bam->x] = Bam->who;

    p = Bam->link;
    map[p->y][p->x] = p->who;
    p = p->link;
    map[p->y][p->x] = p->who;
}
```

이 함수는 게임 스테이지에 뱀을 생성하는 역할을 한다.

‘Something’ 클래스의 객체 ‘Bam’을 생성한다. 이는 뱀의 몸통이 되며, 이후 Something 클래스의 포인터 ‘p’를 생성하여 Something 클래스의 객체로 생성해 준 뱀의 머리와 꼬리를 연결한다. 각각의 객체가 생성될 때에는 인자로 들어간 위치가 함께 설정된다. 생성 후에는 ‘map’ 배열에 뱀의 몸통과 머리를 표시하여 게임 스테이지에 뱀을 생성한다.

- void moveSnake(); 개발자 : 김윤희

```
void Stage::moveSnake()
{
    if (map[Bam->y][Bam->x] != WALL) //뱀의 머리의 현재위치가 벽이 아닌 경우.
        map[Bam->y][Bam->x] = EMPTY; // 해당 위치를 빈공간으로 설정
    Something *q = Bam;
    Something *p = q->link;
    while (p->link != NULL) //뱀의 몸통을 따라 노드를 이동, 각 노드의 좌표를 이전 좌표로
    업데이트
    {
        q->x = p->x;
        q->y = p->y;
        q = p;
        p = p->link;
    }
    if (dir == LEFT) //현재 방향에 따라 뱀의 머리와 머리 다음 노드의 좌표 업데이트
    {

```



```
map[p->y][p->x] = q->who;
q->x = p->x;
q->y = p->y;
p->x--;
}
else if (dir == UP)
{
    map[p->y][p->x] = q->who;
    q->x = p->x;
    q->y = p->y;
    p->y--;
}

else if (dir == RIGHT)
{
    map[p->y][p->x] = q->who;
    q->x = p->x;
    q->y = p->y;
    p->x++;
}

else if (dir == DOWN)
{
    map[p->y][p->x] = q->who;
    q->x = p->x;
    q->y = p->y;
    p->y++;
}

//뱀의 몸통
if (map[p->y][p->x] == WALL || map[p->y][p->x] == SNAKE_BODY) //뱀이 벽이나 다른
{
    map[p->y][p->x] = IMMUNE_WALL;
    gameOver();
}
if (map[p->y][p->x] == GATE)
{
    enterGate(p);
}
if (map[p->y][p->x] == MAGIC_GATE)
{
    enterMagicGate(p);
}
if (map[p->y][p->x] == MAGIC_EXIT)
{
    enterMagicGate(p);
}
```



```

}
if (map[p->y][p->x] == GROWTH_ITEM) {
    eatItem(GROWTH_ITEM);
}
if (map[p->y][p->x] == POISON_ITEM) {

    eatItem(POISON_ITEM);
}
if (map[p->y][p->x] == SPEED_SLOW) {
    eatItem(SPEED_SLOW);
}
map[p->y][p->x] = p->who;
}

```

이 함수는 게임에서 뱀이 이동하는 로직을 관리한다. 뱀의 이동방향에 따라 뱀의 좌표를 업데이트 하고, 뱀이 벽이나 아이템, 또는 뱀 자신의 몸과 충돌하는 경우 해당 상황을 처리한다. 주요 동작으로는,

1. 뱀의 머리가 현재 벽에 있지 않으면 그 위치를 빈 공간으로 설정한다.
2. 뱀의 이동 효과를 주기 위해 뱀의 몸통을 따라가며 각 부분의 위치를 이전 부분의 위치로 업데이트 한다.
3. 현재 이동 방향에 따라 뱀의 머리와 그 다음 부분의 위치를 업데이트 한다.
4. 이동 후, 뱀의 머리가 벽이나 뱀의 몸통에 충돌하면, 그 위치를 immune_wall로 설정하고, gameOver() 함수를 호출한다.
5. 뱀의 머리가 gate에 도달하면, enterGate(p) 함수를 호출하여 gate를 통과하는 로직을 처리한다.
6. 뱀의 머리가 growthitem이나 posionitem, speedslow에 도달하면, eatItem() 함수를 호출하여 아이템을 먹는 로직을 처리한다.
7. 모든 처리가 끝난 후, 뱀의 머리의 위치를 업데이트 한다.

즉, 뱀의 이동과 그에 따른 다양한 상황을 처리하여 게임의 중요한 역할을 한다.

- void enterGate(Something *head); 개발자 : 임혜진

```

void Stage::enterGate(Something *head)
{
    chkEnter = TRUE;
    if (gate1->x == head->x && gate1->y == head->y)
    {
        if (gate2->x == 0)
        {
            head->x = 1;
            head->y = gate2->y;

```





```
        dir = RIGHT;
    }
    else if (gate2->x == COL_END)
    {
        head->x = COL_END - 1;
        head->y = gate2->y;

        dir = LEFT;
    }
    else if (gate2->y == 0)
    {
        head->x = gate2->x;
        head->y = 1;
        dir = DOWN;
    }
    else if (gate2->y == ROW_END)
    {
        head->x = gate2->x;
        head->y = ROW_END - 1;
        dir = UP;
    }
    findRoot(gate2);
    if (dir == LEFT)
    {
        head->x = gate2->x - 1;
        head->y = gate2->y;
    }
    else if (dir == UP)
    {
        head->x = gate2->x;
        head->y = gate2->y - 1;
    }
    else if (dir == RIGHT)
    {
        head->x = gate2->x + 1;
        head->y = gate2->y;
    }
    else if (dir == DOWN)
    {
        head->x = gate2->x;
        head->y = gate2->y + 1;
    }
}
else if (gate2->x == head->x && gate2->y == head->y)
```



```
{
    if (gate1->x == 0)
    {
        head->x = 1;
        head->y = gate1->y;
        dir = RIGHT;
    }
    else if (gate1->x == COL_END)
    {
        head->x = COL_END - 1;
        head->y = gate1->y;
        dir = LEFT;
    }
    else if (gate1->y == 0)
    {
        head->x = gate1->x;

        head->y = 1;
        dir = DOWN;
    }
    else if (gate1->y == ROW_END)
    {
        head->x = gate1->x;
        head->y = ROW_END - 1;
        dir = UP;
    }
    findRoot(gate1);
    if (dir == LEFT)
    {
        head->x = gate1->x - 1;
        head->y = gate1->y;
    }
    else if (dir == UP)
    {
        head->x = gate1->x;
        head->y = gate1->y - 1;
    }
    else if (dir == RIGHT)
    {
        head->x = gate1->x + 1;
        head->y = gate1->y;
    }
    else if (dir == DOWN)
```

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	김혜민	
	Confidential Restricted	Version 1.3	2023-6-18

```

    {
        head->x = gate1->x;
        head->y = gate1->y + 1;
    }
}

stat[3]++;
}

```

이 함수는 뱀이 게임에서 gate를 통과하는 로직을 관리한다. 뱀의 머리가 gate에 도달했을 때, 뱀의 머리 위치와 이동 방향을 업데이트 하고, 통과한 게이트에 따른 처리를 한다.

1. chkEnter = True;로 gate를 통과한 것을 표시한다.
2. 뱀의 머리가 첫 번째 gate에 도달했는지 확인 후, 도달했다면 뱀의 머리는 두번째 gate로 이동하게 되며 이동 후, 뱀의 방향은 gate가 있는 벽의 위치에 따라 결정된다.
3. 만약 뱀의 머리가 두 번째 gate에 도달했다면, 뱀의 머리는 첫 번째 gate의 위치로 이동하게 되며, 뱀의 방향은 gate가 있는 벽의 위치에 따라 결정하게 된다.
4. findRoot(gate2)또는 findRoot(gate1) 함수를 호출하여 gate를 통과한 후 뱀의 머리가 어느 방향으로 이동해야 할지를 결정한다.
5. 뱀의 방향에 따라 뱀의 머리 위치를 업데이트 후 gate를 통과한 횟수를 증가시킨다.

즉, 게임의 gate 통과 로직을 구현하는데 사용되며, 뱀이 gate를 통과하면서 발생하는 다양한 상황을 처리한다.

- void enterMagicGate(Something *head); 개발자 : 임혜진

```

void Stage::enterMagicGate(Something *head)
{
    chkMagicEnter = TRUE;
    if (magicGate1->x == head->x && magicGate1->y == head->y)
    {
        if (magicGate2->x == 0)
        {
            head->x = 1;
            head->y = magicGate2->y;
            dir = RIGHT;
        }
        else if (magicGate2->x == COL_END)
        {
            head->x = COL_END - 1;
            head->y = magicGate2->y;
            dir = LEFT;
        }
    }
}

```




```
else if (magicGate2->y == 0)
{
    head->x = magicGate2->x;
    head->y = 1;
    dir = DOWN;
}
else if (magicGate2->y == ROW_END)
{
    head->x = magicGate2->x;
    head->y = ROW_END - 1;
    dir = UP;
}
findRoot(magicGate2);
if (dir == LEFT)
{
    head->x = magicGate2->x - 1;
    head->y = magicGate2->y;
}
else if (dir == UP)
{
    head->x = magicGate2->x;
    head->y = magicGate2->y - 1;
}
else if (dir == RIGHT)
{
    head->x = magicGate2->x + 1;
    head->y = magicGate2->y;
}
else if (dir == DOWN)
{
    head->x = magicGate2->x;
    head->y = magicGate2->y + 1;
}
}
else if (magicGate2->x == head->x && magicGate2->y == head->y)
{
    if (magicGate1->x == 0)
    {
        head->x = 1;
        head->y = magicGate1->y;
        dir = RIGHT;
    }
    else if (magicGate1->x == COL_END)
    {
```



```
        head->x = COL_END - 1;
        head->y = magicGate1->y;
        dir = LEFT;
    }
    else if (magicGate1->y == 0)
    {
        head->x = magicGate1->x;

        head->y = 1;
        dir = DOWN;
    }
    else if (magicGate1->y == ROW_END)
    {
        head->x = magicGate1->x;
        head->y = ROW_END - 1;
        dir = UP;
    }
    findRoot(magicGate1);

    if (dir == LEFT)
    {
        head->x = magicGate1->x - 1;
        head->y = magicGate1->y;
    }
    else if (dir == UP)
    {
        head->x = magicGate1->x;
        head->y = magicGate1->y - 1;
    }
    else if (dir == RIGHT)
    {
        head->x = magicGate1->x + 1;
        head->y = magicGate1->y;
    }
    else if (dir == DOWN)
    {
        head->x = magicGate1->x;
        head->y = magicGate1->y + 1;
    }
    }
    stat[3]++;
}
```

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	김혜민	
	Confidential Restricted	Version 1.3	2023-6-18

이 함수는 뱀이 게임에서 Magicgate를 통과하는 로직을 관리한다. enterGate()와 유사하게 동작한다. 뱀의 머리가 Magicgate에 도달했을 때, 뱀의 머리 위치와 이동 방향을 업데이트 하고, 통과한 Magic Gate에 따른 처리를 한다.

- int findRoot(Something *gate); 개발자 : 김윤희

```
int Stage::findRoot(Something *gate)
{
    for (int i = 0; i < 4; i++)
    {
        if (dir == LEFT)
        {
            if (map[gate->y][gate->x - 1] == EMPTY)
                return dir;
            else
                dir = KEY_UP;
        }
        else if (dir == KEY_UP)
        {
            if (map[gate->y - 1][gate->x] == EMPTY)
                return dir;
            else
                dir = RIGHT;
        }
        else if (dir == RIGHT)
        {
            if (map[gate->y][gate->x + 1] == EMPTY)
                return dir;
            else
                dir = DOWN;
        }
        else if (dir == DOWN)
        {
            if (map[gate->y + 1][gate->x] == EMPTY)
                return dir;
            else
                dir = LEFT;
        }
    }
    return dir;
}
```



```
}

```


이 함수는 뱀이 gate를 통과한 후 다음으로 이동해야 할 방향을 찾는 함수이다. 이 함수는 gate 위치 주변을 돌아보며 빈 공간을 찾고 이때 체크하는 순서는 뱀의 이동 방향(dir)에 따라 달라진다.

1. dir == left인 경우 gate 왼쪽의 빈 공간을 찾고, 없으면 다음 방향 up으로 이동한다.
2. dir == up인 경우, gate 위쪽의 빈 공간을 찾고, 없으면 다음 방향 right로 이동한다.
3. dir == down인 경우, gate 아래쪽의 빈 공간을 찾고, 없으면 다음 방향 left로 이동한다.
4. 만약 모든 방향에서 빈 공간을 찾지 못했다면, 최종적으로 뱀의 현재 방향을 반환한다.

즉, 뱀이 gate를 통과한 후 안전하게 이동할 수 있는 방향을 찾으며 뱀이 벽이나 자신에 몸에 부딪히는 등의 문제를 피할 수 있다.

- void eatItem(int item); 개발자 : 김윤희

```
void Stage::eatItem(int item)
{
    if (item == GROWTH_ITEM)
    {
        if (stat[0] == 10)
            return;
        Something *p = new Something(Bam->y, Bam->x, SNAKE_BODY);
        if (Bam->x - Bam->link->x == 1)
            p->x++;
        else if (Bam->y - Bam->link->y == 1)
            p->y++;
        else if (Bam->x - Bam->link->x == -1)
            p->x--;
        else if (Bam->y - Bam->link->y == -1)
            p->y--;
        p->link = Bam;
        Bam = p;
        if (map[Bam->y][Bam->x] != WALL)
            map[Bam->y][Bam->x] = Bam->who;
        stat[0]++;
        stat[1]++;
    }
    else if (item == POISON_ITEM)
    {
        map[Bam->y][Bam->x] = EMPTY;
        Bam = Bam->link;
        stat[0]--;
    }
}
```

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	김혜민	
	Confidential Restricted	Version 1.3	2023-6-18

```

        stat[2]++;
    }
    else if(item == SPEED_SLOW)
    {
        timeoutMs+=100;
        stat[4]++;
    }
}

```

이 함수는 뱀이 아이템을 먹었을 때를 처리하는 함수이다. growth_item과 poison_item을 먹었을 때 두 가지 경우를 처리한다. 주요 동작으로는,

1. growth_item의 경우
 - 뱀의 현재 길이가 10이라면 더 이상 성장하지 않으므로 함수를 종료한다. 그렇지 않다면 something노드를 생성하고 이를 뱀의 몸통에 추가하고 이는 뱀의 성장을 의미한다.
 - 새로운 노드의 위치는 뱀의 현재 머리 위치를 기준으로 계산된다. 새 노드의 위치는 뱀의 이동 방향에 따라 다르며, 뱀이 오른쪽으로 이동 중이라면 새로운 노드는 뱀의 왼쪽에 추가된다.
 - 뱀의 길이와 먹은 성장 아이템 수가 증가한다.
2. poison_item의 경우
 - 뱀의 머리가 있는 위치의 맵을 empty로 설정한다.
 - 뱀의 머리를 다음 노드로 이동시킨다. 이는 뱀의 길이가 줄어듦을 의미한다.
 - 뱀의 길이는 감소하고, 먹은 독 아이템 수는 증가한다.
3. speed_slow의 경우
 - 현재 뱀의 속도가 감소한다.

즉, 뱀이 아이템을 먹었을 때 게임의 상태를 적절하게 업데이트 한다.


- void setMission(); 개발자 : 임혜진

```

void Stage::setMission()
{
    finish = chkEnter = FALSE;
    memset(stat, 0, sizeof(stat));
    memset(statMission, 0, sizeof(statMission));
    memset(chkMission, ' ', sizeof(chkMission));

    statMission[0] = 5;
    statMission[1] = 2;
    statMission[2] = 2;
}

```


 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	김혜민	
	Confidential Restricted	Version 1.3	2023-6-18

```

statMission[3] = 2;
}

```

이 함수는 미션을 설정하는 함수이다. 게임 레벨이 시작될 때 호출되어 각 레벨에 대한 목표를 설정한다. 주요 동작으로는,

1. finish와 chkEnter라는 두개의 변수를 false로 설정한다.
2. memset함수를 사용하여 현재상태를 나타내는 stat, 각 레벨의 미션 목표를 나타내는 statMission, 미션의 완료 여부를 나타내는 chkMission 배열들의 모든 요소를 초기화한다.
3. 각 미션의 목표는 뱀의 길이는 5, growthitem과 poisonitem은 2, gate 통과 목표도 2로 설정된다.

즉, 각 레벨에 대한 미션을 설정, 게임의 현재 상태와 미션 완료 상태를 초기화 하는 역할을 한다.

- bool isMissionClear(); 개발자 : 임혜진


```

bool Stage::isMissionClear()
{
    int count = 0;
    for (int i = 0; i < 4; i++)
    {
        if (stat[i] >= statMission[i])
        {
            chkMission[i] = 'v';
            count++;
        }
        else if (!i)
            chkMission[i] = ' ';
    }
    if (count == 4)
        return TRUE;
    return FALSE;
}

```

이 함수는 게임의 미션이 완료되었는지를 판단하는 함수이다.

1. 미션 완료 항목을 나타내는 count 라는 변수를 0으로 초기화한다.
2. 반복문을 통해 4개의 미션 항목을 확인한다.
3. 만약 각 항목에 대한 현재 성과가 목표치보다 크거나 같다면 해당 항목이 완료된 것으로 간주하여 'v'로 설정하고 count를 1증가시킨다.
4. 모든 미션 항목을 확인한 후, count가 4라면 true를 반환하고 그렇지 않다면 false를 반환한다.

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	김혜민	
	Confidential Restricted	Version 1.3	2023-6-18

즉, 모든 미션 항목이 완료되었는지를 확인하고 그 결과를 반환한다. 이는 게임 진행 상황을 체크하고 미션 클리어 조건을 평가하는데 사용된다.

- void Gameover(); 개발자 : 임혜진

```
void Stage::gameOver()
{
    finish = true;
}
```

이 함수는 게임 오버가 되면 finish 변수를 true로 변경한다.

- void alert(int posY, int posX, const string msg, bool stopFlag); 개발자 : 신수민

```
void Stage::alert(int posY, int posX, const string msg, bool stopFlag)
{
    WINDOW *alert = newwin(7, msg.length() * 2, posY, posX);
    box(alert, 0, 0);

    watttrn(alert, COLOR_PAIR(0));
    wbkgd(alert, COLOR_PAIR(2));
    mvwprintw(alert, 3, msg.length() / 2, msg.c_str());
    wrefresh(alert);
    if (!stopFlag)
        usleep(1750000);
}
```

이 함수는 게임 스테이지에 경고 메시지를 표시하는 역할을 한다.

newwin() 함수를 사용하여 alert라는 새로운 윈도우를 생성한다. 경고 메시지(게임오버, 스테이지 클리어, pause 작동 시)를 1.75초 동안 화면에 표시한다.

[Main.cpp]

- int main() 개발자 : 김윤희, 신수민, 임혜진

```
#include "Stage.h"
```



```
int main()
{
    Stage view = Stage();
    string game;
    while(1)
    {
        game = view.menu();
        clear();
        if(game == "PLAY")
            view.play();

        else if(game == "HELP")
            view.help();
        else
            break;
    }
    return 0;
}
```

이 함수는 게임의 메인 실행 흐름을 제어한다. 게임의 시작점으로 게임의 각 부분을 연결하고 실행 순서를 제어한다. 주요 동작으로는

1. Stage 클래스의 객체는 view를 생성하여 모든 stage와 관련된 작업을 제어한다.
2. game이라는 문자열 변수를 선언하여 사용자가 선택하는 메뉴 옵션을 저장하는데 사용한다.
3. 무한루프를 생성하여 사용자가 게임을 종료하기 전까지 반복한다.
4. 사용자의 선택에 따라 메소드를 호출하고, exit를 선택했다면 무한 루프를 빠져나가게 된다.
5. return 0;으로 메인함수가 성공적으로 종료되었음을 운영 체제에 알리고 정상적으로 종료한다.

2.2.3 활용/개발된 기술


작성요령 (10점)

프로젝트 수행에 사용한 외부 기술/라이브러리를 나열하여 작성한다. 각각 기술을 이 프로젝트에 적용할 때, 도움 받거나 해결하고자 하는 기능에 대해 상세히 설명한다.

NCURSES / STL 라이브러리 등을 포함하여 설명한다.

또한, 이 프로젝트를 수행하면서, 새롭게 고안한 알고리즘 등이 있다면 설명한다.

- Ncurses library: window를 띄우고, 화면에 무언가를 띄우고, 효과를 적용시키는 등을 위해서 사용했습니다.

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	김혜민	
	Confidential Restricted	Version 1.3	2023-6-18


- STL: STL의 라이브러리의 Sequence Container에 속하는 Vector를 사용했습니다. Vector의 원소 타입을 pair로 받아서 좌표 위치를 가리킬 수 있도록 하였습니다.
- cstring: cstring 헤더 안에 있는 memset 함수는 배열의 메모리 블록을 특정 값으로 초기화해 줍니다. 이를 사용하여 Mission Board와 Score Board의 배열의 크기를 초기화해 주었습니다.
- cstdlib: cstdlib 헤더 안에 존재하는 rand 함수와 srand 함수를 사용하여 Gate와 Item을 랜덤 좌표에 생성될 수 있도록 만들었습니다.
- unistd.h: unistd 헤더 안에 존재하는 usleep 함수를 이용했습니다. 이 함수로는 지정된 ms만큼 프로그램을 멈춰 뒀다가 실행시킬 수 있었습니다.

2.2.4 현실적 제한 요소 및 그 해결 방안

작성요령 (5점)

제안된 프로젝트의 단계 별 수행에 있어, 제한 요소를 찾아 작성한다. 해당 제한 요소를 해결하기 위해서 어떤 방법으로 해결하였는지 작성한다.

- 1 ~ 3단계: map 생성 후 map 위에 snake를 어떻게 출력해야 할지에서 막혔었습니다. 하지만 vector를 사용하여 좌표를 지정해 map 위에 snake를 각각 출력하면 된다는 것을 깨닫고 고쳐 나갔습니다. 이후 map 위에 snake를 출력해야 할 때 모두 vector 방식으로 할 수 있게 되었습니다. 또 뱀과 아이템을 생성하려 할 때, 생각보다 많은 부분을 신경써야 해서 많이 해맸습니다. grow, poison, speedslow 아이템들을 구현하면서 윈도우에 나타나는 +, - 등의 기호를 어떻게 표현해야 할 지 많이 고민했습니다. 고민 끝에 const string 타입의 itemindex 배열을 생성하여 각 인덱스를 변수의 상수로 지정하여 뱀의 머리와 꼬리, 그리고 각 아이템을 표시하도록 하였습니다.
또한 뱀의 speed를 stage가 달라짐에 따라 변화를 주는 것을 설정할 때 어떤 방식으로 구현해야 할 지 고민을 많이 했습니다. 고민 끝에 저는 speed가 담긴 배열 speedMs 를 선언하고 각 stage별로 초기화 할 값들을 저장하였습니다. speed라는 정수형 변수를 사용해 그 배열의 인덱스를 조정할 수 있도록 하였고, 그 값을 timeoutMs 변수에 주고 이 변수를 timeout함수에 인자로 주어 delay되는 효과를 가져와 속도를 조정하였습니다. 하지만 speedslow 아이템에서는 speed변수를 조정하면 현재 속도가 아니라 다음 게임에 speed가 줄어들어 speed변수를 조정하지 않고 바로 timeoutMs값을 조정하여 현재속도의 변화를 주었습니다. 또한 가시적으로 편하게 해석하기 위해 speed++ 하면 속도가 증가되는 것을 나타내기 위해 speedMs의 배열을 내림차순으로 저장하였습니다.
- 4단계: gate 통과 후 snake의 방향 설정이 어려웠으나, 먼저 출구 쪽 벽이 어느 방향에 있는지 체크했습니다. 이후, snake의 head 방향을 탐색 후 비교하여 snake의 이동 경로를 설정할 수 있었습니다. 또 Magic Gate를 추가로 생성하고 실행시켰을 때, 하나의 게이트를 통과하면 다음

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	김혜민	
	Confidential Restricted	Version 1.3	2023-6-18

게이트를 통과할 때 에러가 발생하였습니다. 이 에러의 원인을 오랫동안 찾았는데 기존 Gate와 겹치는 변수가 있어 덮어 씌워지는 문제였습니다. 새로운 메소드와 변수들을 더 만들어주어서 이 문제를 해결할 수 있었습니다.

- 5단계: mission board와 score board 안 변수들의 값을 일일이 바꾸어 준다는 점에서 번거롭고 어떤 방식을 사용해야 할지 막혔었습니다. 하지만 배열을 사용하여 쉽고 메모리 사용이 덜하게 해결할 수 있었습니다.

2.2.5 결과물 목록

작성요령 (5점)

결과물 목록을 작성한다. 목록은 제출하는 파일과 각 파일의 역할을 간략히 설명한다.

- Something.h
: 뱀이나 게이트 등의 생성과 동작을 위한 Something 구조체가 정의되어 있다.
- Stage.h
: ncurses라이브러리, iostream, cstring, cstdlib, unistd.h, vector, fstream, something.h 등의 함수 구현에 필요한 헤더파일을 include하는 파일이다. 또한 게임 구현에 필요한 상수들을 선언하고 class stage를 포함한다. class stage안에서 stage생성자를 호출하고, 게임에서 필요한 메소드들을 public과 private로 선언한다.
- Stage.cpp
Stage.h를 include하고 Stage.h에서 선언된 메소드들이 구현되어 있는 파일이다. 게임 진행에 있어서 가장 핵심인 부분이라고 할 수 있다.
- Main.cpp
stage class의 객체를 생성하고 그 객체를 통하여 게임의 시작과 종료를 책임진다.
- Makefile
위의 파일들을 한꺼번에 compile하여 실행시킬 수 있는 실행 파일이다.



3 자기평가

작성요령 (5점)


프로젝트를 수행한 자기 평가를 서술한다. 팀원 개개인의 자기 평가가 포함되어야 하며, 본인의 역할, 프로젝트 수행 시 어려운 점, 도움이 되었던 점, 이 프로젝트 운영에 개선이 필요하다고 생각하는 점을 충분히 서술한다.

김윤희

: SnakeGame 과제를 맞닥뜨렸을 때는 설렘이 컸지만 정말 프로젝트를 시작하려고 보니 생각보다 많이 막막했습니다. 평소 C++ 언어를 공부하였지만 하나의 프로젝트를 수행한 경험이 없어 ncurses라이브러리 사용, makefile 만드는 법 등 외에도 사소한 곳에서 어려움을 많이 느꼈습니다. 하지만 검색도 해보고, 도움도 요청하면서 단계별로 해결하다보니 뿌듯함을 원동력으로 삼아 쪽 개발을 진행할 수 있었습니다. 또한 개발 진행과정을 눈으로 보면서 확인하고, 문제점을 찾고 해결하면서 좌절감과 희열을 동시에 느낄 수 있었습니다. 수업시간에 배운 이론적 내용을 실제로 제가 작성해보고 수행하고 경험하면서 온전히 저의 것으로 만든 느낌이 들어 매우 의미있었지만 아직 미흡한 부분이 많아 팀원들의 도움과 조언을 받으며 수행한 부분도 많았습니다. 팀원들과 소통하고 협력하며 협동심도 기르고, 저의 부족한 점도 깨닫고, 프로젝트 수행에 대한 뿌듯함도 느끼며 여러 방면에서 성장할 수 있는 유익하고 의미있던 과제였습니다.

신수민

: 처음 과제가 주어졌을 때, C++ 언어로 Snake Game을 만드는 것에 두려움이 많았습니다. 여러 프로그래밍 언어 중 C++는 자신이 없는 언어 중 하나였기 때문이라는 이유였습니다. 그래도 팀원들이 능력이 대단하신 분들이라서 팀을 짜고 난 후에는 걱정이 줄었습니다. 게임 구현을 위해 첫 회의를 하는 중 느낀 것은, 이 게임 구현의 세부 사항을 계획하는 단계까지도 그렇게 어렵지는 않았다는 것입니다. 하지만 다른 팀원들과 저 사이에 회의라는 소통을 할 수 있는 시간을 만드는 것에 있어서 저와 다른 팀원들은 시간이 너무나도 맞지 않았고 특히 이 부분에서는 스트레스를 많이 받았습니다. 그리고 개발할수록 다른 팀원들과는 다르게 구현에 어려움을 많이 겪은 저는 보고서에 적힌 것처럼 실제로 구현을 많이 말아서 하지는 못했습니다. 제가 맡은 부분들은 다른 부분의 로직을 구현하는 것보다는 제가 느끼기에는, 비교적으로 너무나도 쉽고 간단한 편이었습니다. Snake가 움직이는 것 관련한 부분을 구현하고 싶었지만, 괜한 욕심이라고 생각했고 효율적인 팀 활동을 위해서는 저보다 다른 팀원이 하는 것이 맞다고 생각했습니다. 이후, 추가 wall 사항인 Magic Gate를 구현하는 것에도 도전했었습니다. 하지만 몇 시간씩 밤새우며 연달아서 매달려 하는데도 구현하는 것에 너무 많은 어려움과 스트레스를 겪었습니다. 결국 막힌 부분을 해결하지 못해 다른 팀원이 구현하게 되었습니다. 그래서 팀원들에게 조금 미안하기도 했고, 저에게 화도 많이 났습니다. 이 부분은 종강 후에 다시 보며 뭐가 문제인지 차근차근 살펴보고 싶습니다. 짧다면 짧고, 길다면 긴 시간을 Snake Game에 투자하면서 이 분야에서 계속 나아가는 것이 맞나 고민도 많이 했었지만, 다른 팀원들이 많이 고생한 결과물을 보면서 느낀 것은 희열이었습니다. 제가 앞으로 이 분야에 나아가며 저를 발전시키는 데에 있어서는 굉장히 의미 있는 시간이었다고 생각합니다.


 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	김혜민	
	Confidential Restricted	Version 1.3	2023-6-18

임혜진

: 지금까지는 예제를 활용하거나 알고리즘 문제를 풀면서 코드를 짜고, 터미널에 찍히는 출력값을 보는 것이 다였는데, 직접 아이디어를 내고 구현하여 게임을 만들어본 것은 처음이었습니다. 프로젝트를 안내받았을 때는 정말 막막하고 어려워서 시간 내에 잘 할 수 있을지 걱정이 많았는데, 하다보니 조금씩 감이 잡히면서 더 속도가 붙었던 것 같습니다. 생각한대로 기능이 잘 동작하지 않을 때도 정말 많았지만, 안되는 원인을 찾아냈을 때는 정말 기뻐했습니다. 또한 팀프로젝트였기에 더 좋았던 것 같습니다. 누군가 하다가 잘 작동하지 않는 부분이 생기면 다른 팀원들과 함께 고민하며 해결해 나갈 수 있었습니다. 또 추가 사항을 구현하기 위해 아이디어 회의를 여러차례 진행했었는데, 다양한 아이디어가 나와서 흥미로웠습니다. 혼자 진행한 프로젝트였다면 아이디어도 부족했을 것 같다고 느꼈습니다. 팀프로젝트와 게임 개발을 동시에 경험해볼 수 있었기에 유익한 프로젝트였던 것 같습니다.

4 참고 문헌

번호	종류	제목	출처	발행년도	저자	기타
1	웹페이지	c++ reference	https://cplusplus.com/reference/			
2	웹페이지	[linux] NCURSES란?	https://anythink.tistory.com/entry/Linux-NCURSES-프로그래밍	2013	투명잉크	
3	웹페이지	[Linux] Ncurses	https://velog.io/@dev-hoon/koxzrs13	2022	정재훈	

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	김혜민	
	Confidential Restricted	Version 1.3	2023-6-18

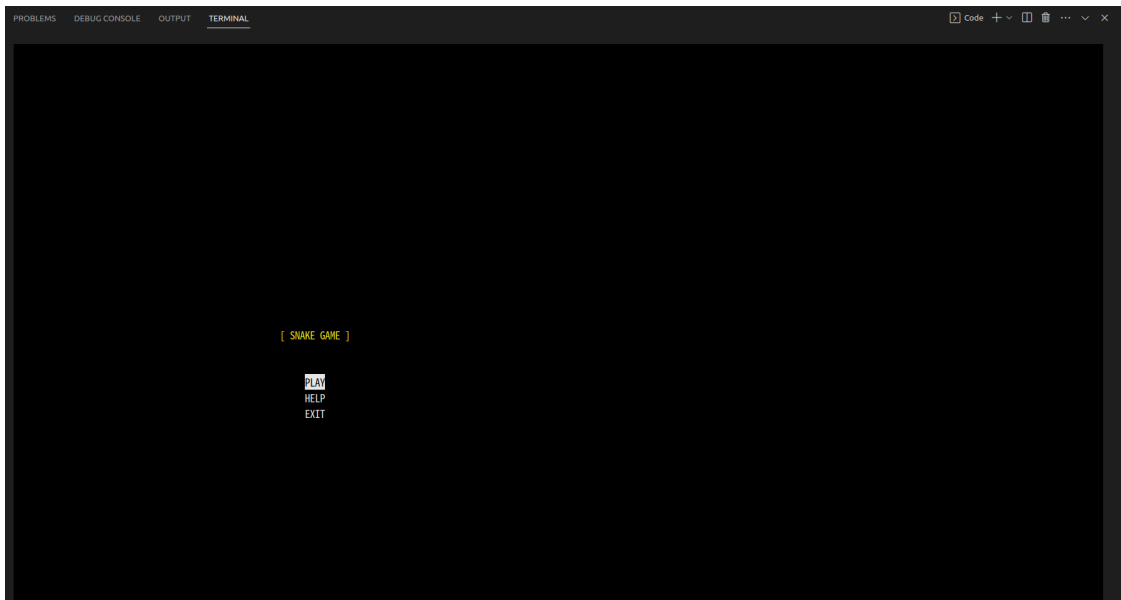
5 부록

작성요령 (15점)

프로젝트의 결과물을 사용하기 위한 방법에 대해서 작성하세요.

5.1 사용자 매뉴얼


- 시작 화면



→ 파일을 실행시키면 Snake Game이 시작되기 전에 위와 같은 화면이 나타납니다. 이때 사용자는 방향키(위, 아래)를 통해 PLAY, HELP, EXIT 중 하나를 선택할 수 있습니다.

- Game play 화면

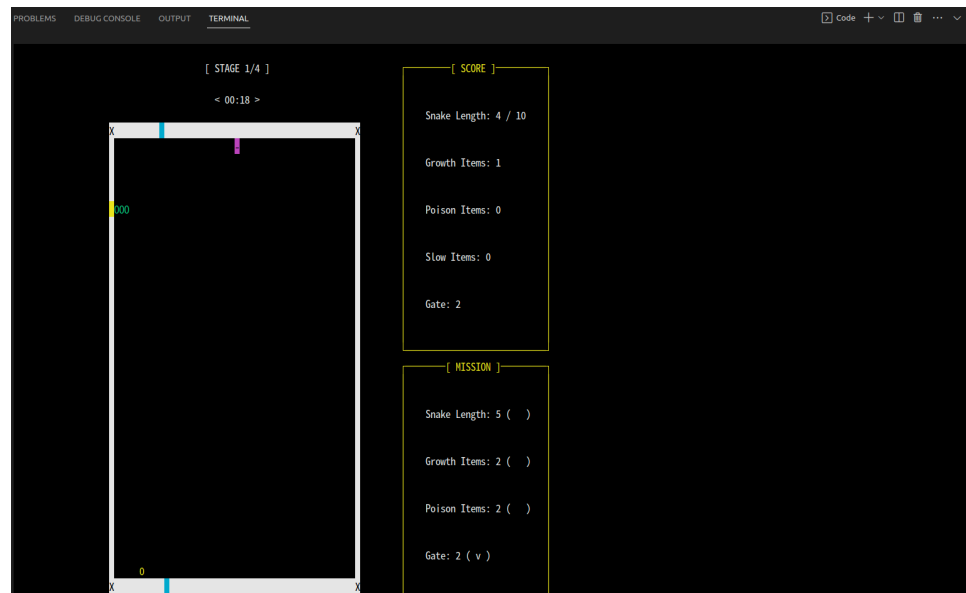
- 일반 Gate를 통과하는 화면

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	김혜민	
	Confidential Restricted	Version 1.3	2023-6-18




→ Snake가 Gate를 통과하게 되면, 우측의 Mission Board에 통과한 Gate의 개수가 기록됩니다.

- Magic Gate를 통과하는 화면



→ Snake가 Magic Gate를 통과해도 마찬가지로, 우측의 Mission Board에 통과한 Gate의 개수가 기록됩니다. 그리고, 화면에 보이는 것과 같이 Magic Gate는 들어가는 gate는 보이지만 나오는 gate는 보이지 않습니다.

- Growth Item을 획득하는 장면

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	김혜민	
	Confidential Restricted	Version 1.3	2023-6-18




→ Snake가 Growth Item을 획득하게 되면, 우선 Snake의 길이가 하나 늘어나게 됩니다. 그리고, 우측의 Scoreboard에 획득한 Growth Item의 개수가 기록됩니다. 또한, 우측의 Snake length가 1 늘어나게 됩니다.

- Poison Items를 획득하는 장면



→ Snake가 Poison Item을 획득하게 되면, 우선 Snake의 길이가 하나 줄어들게 됩니다. 그리고, 우측의 Scoreboard에 획득한 Poison Item의 개수가 기록됩니다. 또한, 우측의 Snake Length가 1 줄어들게 됩니다.

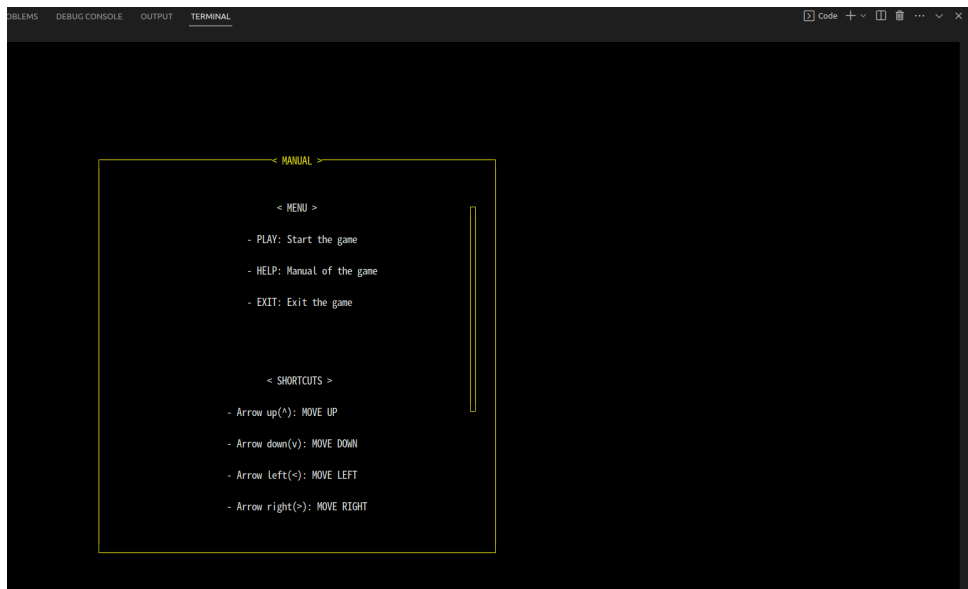
 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	김혜민	
	Confidential Restricted	Version 1.3	2023-6-18


- Slow Item을 획득하는 장면

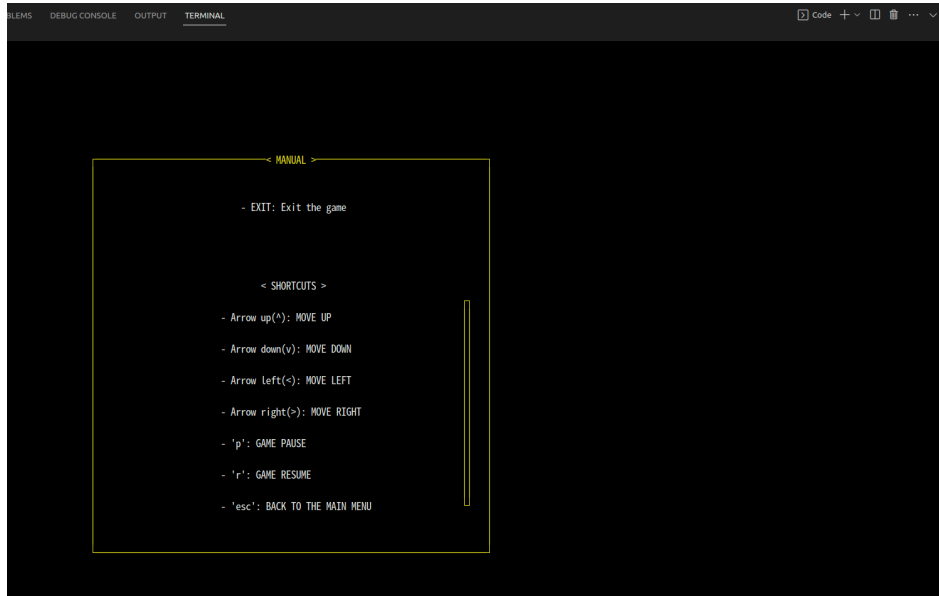


→ Snake가 Slow Item을 획득하게 되면, 우선 Snake의 속도가 한 단계 느려집니다. 또한, 우측의 Score board에 획득한 Slow Item의 개수가 기록됩니다.

- HELP 화면



 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	김혜민	
	Confidential Restricted	Version 1.3	2023-6-18




→ 시작 화면에서 사용자가 방향키를 이용해 HELP를 선택하면, 이러한 화면이 나타납니다. HELP 화면에서는 시작 화면에서의 메뉴 세 개가 가리키는 메뉴얼을 나타내고 있습니다. 그리고 방향키의 방향을 명시해 주고 있습니다. 마지막으로, 게임 도중에 게임을 정지시킬 수 있는 단축키와 정지 후 재시작할 수 있게 해 주는 단축키를 소개해 주고 있습니다. 또한, 메인 메뉴로 돌아갈 수 있게 해 주는 단축키도 소개해 주고 있습니다.

- EXIT 화면



→ 시작 화면에서 사용자가 방향키를 이용해 EXIT를 선택하면, 게임에서 나가도록 해 줍니다.

 국민대학교 소프트웨어학부 C++프로그래밍	결과보고서		
	프로젝트 명	Snake Game	
	팀 명	김혜민	
	Confidential Restricted	Version 1.3	2023-6-18

5.2 설치 방법

1. Ncurses 라이브러리를 설치합니다
2. 터미널에 make를 입력하면 실행파일 run이 생성됩니다.
3. 터미널에서 ./run 을 입력하면 게임이 실행됩니다.

```

문제  출력  디버그 콘솔  터미널

imagine@imhyejin-ui-MacBookPro snakegame % cd step5
imagine@imhyejin-ui-MacBookPro step5 % make
g++ -g -std=c++11 -c Stage.cpp
g++ -g -std=c++11 -o run Stage.o Main.o -lncurses
○ imagine@imhyejin-ui-MacBookPro step5 % ./run

```